Fiche TD14 : Algorithmique du texte

Exercice 3 *Autour de l'algorithme de Rabin-Karp*

On considère un texte n'utilisant que des caractères ASCII. Le haché d'une chaîne $c_0...c_k$ de caractères est calculé à l'aide de la fonction de hachage h définie comme suit : on commence par calculer $\sum_{i=0}^{k} n_i r^{k-i}$ où n_i est le code ASCII de c_i (donc appartient à $[\![0, 255]\!]$) et $r = 2^8$. Puis on réduit le tout modulo un nombre premier q.

1. Si l'on connait $h(c_0...c_k)$ expliquer comment calculer $h(c_1...c_k a)$ où a est une lettre. Discuter de la complexité d'une telle opération.

2. Appliquer l'algorithme de Rabin-Karp à la recherche du motif "program" dans la citation (encore due à Knuth) : "Premature optimization is the root of all evil in programming" dans le cas où $q = 7$. Qu'en penser ?

3. (bonus) On considère une matrice t et un motif en deux dimensions m. On cherche à calculer une position du motif m dans la matrice t si une telle position existe ; par exemple :

$$\text{Le motif } \begin{pmatrix} A & U \\ L & E \end{pmatrix} \text{ apparaît à la position } (1,2) \text{ dans la matrice } \begin{pmatrix} R & A & G & E \\ E & T & A & U \\ C & A & L & E \end{pmatrix}$$

Adapter l'algorithme de Rabin-Karp afin d'accomplir efficacement cette tâche.

Exercice 4 *Codage de Huffman statique*

Deux entités souhaitent échanger un long texte t dont on sait qu'il ne contient que les caractères de l'alphabet $[\![0, 9]\!]$. On sait de plus que la probabilité d'apparition de chacune des lettres de l'alphabet dans t est donnée par le tableau suivant :

Lettre	0	1	2	3	4	5	6	7	8	9
Probabilité d'apparition	0.05	0.1	0.11	0.11	0.15	0.06	0.08	0.2	0.07	0.07

Afin de compresser t, on propose de coder chaque lettre à l'aide d'un code obtenu par l'algorithme de Huffman. Pour ce faire, on remplace simplement la définition du poids d'un arbre de Huffman par : "le poids d'un arbre de Huffman est la somme des probabilités d'apparition de chacune des lettres portées par ses feuilles".

1. Construire l'arbre de Huffman correspondant à cette situation.

2. Indiquer le code obtenu pour chaque lettre de $[\![0, 9]\!]$. A-t-on unicité de la clé de codage (c'est-à-dire, de la fonction associant lettres et codes) obtenue par cette méthode ?

3. Calculer le nombre de bits moyen d'un code pour une lettre de l'alphabet considéré.

On appelle "taux de compression" (sous entendu, d'un algorithme de compression de données) le quotient entre la (taille des données non compressées) et la (taille des données compressées).

4. Quel est le taux de compression moyen obtenu grâce à l'algorithme de Huffman dans la situation proposée sachant que sans compression chaque entier serait représenté sur 8 bits (dans à peu près tous les langages, on stocke un entier sur au moins un byte) ?

Exercice 5 *L'algorithme de Huffman produit une clé de codage minimale*

Si H est un arbre de Huffman pour la chaîne s définie sur un alphabet A, on appelle *coût de* H le coût de la clé de codage qui lui est associé, c'est à dire la somme suivante :

$$\sum_{u \in A} Occ(u) \times c(u)$$

où $Occ(u)$ est le nombre d'occurrences de la lettre u dans la chaîne s et $c(u)$ est le nombre de bits du code pour la lettre u d'après l'arbre de Huffman H. On dit que H est un arbre de Huffman minimal si son coût est minimal.

1. On considère un arbre de Huffman H. L'arbre H' est obtenu à partir de H en permutant les feuilles étiquetées par les caractères u et v dans H.

a) Calculer la différence Δ entre le coût de H' et le coût de H en fonction de $c(u)$, $Occ(u)$, $c(v)$ et $Occ(v)$.

b) Montrer que dans un arbre de Huffman minimal, les feuilles de profondeur maximale contiennent les lettres apparaissant le moins souvent.

2. Considérons un arbre de Huffman H possédant deux feuilles ayant même père n et étiquetées par deux caractères u et v d'occurrences minimales. On construit à partir de H un arbre H' obtenu en enlevant de H les feuilles u et v et en considérant que n est une feuille d'occurence $Occ(n) = Occ(u) + Occ(v)$.

   a) Montrer que si H' est un arbre de Huffman minimal alors il en va de même pour H.

   b) En déduire que l'algorithme de Huffman construit un arbre de Huffman minimal.

Exercice 6 *Compression et décompression LZW*

On considère l'alphabet latin classique majuscule et non accentué dans lequel on code chaque lettre par sa position dans l'alphabet (le code de A est 1, celui de B vaut 2... celui de Z est 26. Le premier code libre est donc 27). Avec ces conventions :

1. Compresser avec l'algorithme LZW le mot : ENGAGELEJEUQUEJELEGAGNE
2. En considérant que chaque lettre est codée sur 8 bits et que chaque nombre codant une lettre l'est aussi, quel est le taux de compression (voir définition à l'exercice 4) obtenu sur ce mot ?
3. Reprendre les deux questions précédentes avec le mot : AAACCAACCACAACCAACAAACA
4. Décompresser avec l'algorithme LZW le texte compressé : 20 1 14 2 28 31 19